# SIMAC – Innovation Company

## Stem Detection in Images Using Computer Vision

## Application of color segmentation and blob analysis in Python for robotic systems

**Technical Report**

**Author: Natalia Hoyos González**

# Introduction

This project is developed as part of an initiative to advance the automation of trees and shrub pruning through the use of intelligent robotic systems. The solution is based on computer vision techniques, specifically in the detection of shapes and colors, with the objective of identifying the location of plant stems using Python.

This report explains in detail the tools used to carry out this development, combining advanced technical concepts with essential fundamentals of image processing and machine vision. It also analyzes how this solution can be applied in different industrial and agricultural contexts and proposes possible improvements and future developments.

# Machine Vision

This section describes the theoretical foundations and technical tools used to develop the visual detection system. The computer vision principles applied in the project are explained, including color segmentation, color space conversion, and detection of regions of interest using blobs (groups of adjacent pixels that form a region with defined visual structure). These techniques allow the system to robustly identify stems in images of shrubs or trees.

The development is supported by four main Python libraries, which facilitate the implementation of computer vision algorithms:

- **matplotlib.pyplot**: Used for image visualization during development and debugging.
- **machinevisiontoolbox**: Provides advanced tools for image processing, including blob detection.
- **numpy**: Enables efficient matrix handling and numerical operations, essential for image data processing.
- **cv2 (OpenCV)**: A fundamental library for image manipulation, color space conversion, mask creation, and morphological operations.

Below, we will detail how these tools work, along with the key computer vision principles applied in the project.

## Feature Extraction

One of the fundamental problems in computer vision applied to robotics is the large amount of data generated by a camera compared to the relatively small amount of information a robot needs to act. For example, a camera can generate tens of millions of pixels per second, while a

typical mobile robot only requires two variables for its control (linear and angular velocity), and an industrial robotic arm needs just six (one for each joint).

This imbalance between available and required data is known as the "information overload problem," and it is addressed through a technique known as **feature extraction**. The objective of this technique is to reduce visual data to its essence: identifying the most relevant information in the image, such as contours, shapes, color regions, or key points, that can be directly useful for the robot's control system. In this project, such extraction is done by segmenting the brown color and subsequently identifying blobs, which represent possible stem locations in the image.

## Color Perception and Processing

Color is a fundamental property in human perception and also a powerful tool in artificial vision systems. In this project, stem detection is primarily based on recognizing its characteristic color (brown tones), which allows it to be segmented from the background using thresholding techniques in the HSV color space.

From a physical standpoint, the color we perceive comes from the interaction between light (illuminance) and the reflective properties of the object. When a light source (illuminant) strikes a surface, part of the energy is reflected. This reflected light (luminance) enters the camera sensor or the human eye. How the object reflects each wavelength depends on its spectral reflectance, which determines the final color we observe.

There are two main types of reflection:

- **Specular reflection**, which occurs on shiny surfaces and reflects light in a precise direction (like a mirror).
- **Diffuse or Lambertian reflection**, typical on matte surfaces, which reflects light in multiple directions. This latter model is more realistic to represent how we see objects like fruits, leaves, or stems.

In computer vision, the use of the **HSV color space** (Hue, Saturation, Value) allows for a better separation between color information (hue) and light intensity. Unlike the RGB space, where values are mixed, HSV facilitates the creation of binary masks that isolate specific colors. In this project, a specific range of hue values is used to identify brown pixels, thus generating a mask that isolates possible stems.

This type of color segmentation is effective as long as lighting conditions are controlled, since luminance can alter pixel values. To improve robustness, techniques such as gamma correction or conversion to chromaticity coordinates can be applied, which help separate color from intensity. However, in this case, a direct and efficient HSV-based implementation was chosen, suitable for semi-controlled environments.

### Blob Detection

Once the image has been segmented to isolate regions of interest (in this case, brown pixels that possibly represent stems), the next step is to identify connected regions that share similar characteristics. These regions are known as **blobs** (Binary Large Objects) and are groupings of contiguous pixels that form a visually coherent shape or object.

In digital image processing, blobs provide a structured representation that allows abstraction of visual content into interpretable elements, such as areas, perimeters, centroids, or contours. This approach is especially useful in robotic vision, as it allows a complex image to be translated into a list of individual objects with measurable properties.

The algorithm used in this project is based on **connected component analysis**: a technique that scans the binary image resulting from segmentation and groups white pixels that are connected. Each group is assigned a unique identifier, allowing one blob to be distinguished from another.

To avoid common errors, such as false blobs caused by noise or specular highlights, **filters for minimum and maximum area** are applied, and those blobs with a zero-order spatial moment ($M_{oo}$) equal to zero are discarded, as they indicate invalid or empty regions.

## Problem Solution Proposal

As mentioned previously, the main objective of the project is to identify the location of the stem within an image using computer vision. To achieve this, the previously described libraries were used along with specific segmentation and region analysis techniques.

This section presents, step by step, how these tools were integrated to build a functional solution. The processing stages are detailed, from image acquisition and preprocessing to obtaining the coordinates of the detected stem, explaining how each system component contributes to the final result. The test image in Figure 1 was used to verify the code.

*Figure 1: Test image to run the code with and observe results.*

## Image Loading and Preprocessing

The process begins by loading the image in RGB format and applying Gaussian smoothing to reduce high-frequency noise. This step is crucial to improve the stability of subsequent segmentation and is performed using the `smooth()` function with a low sigma value (see Figure 2).

```
# Step 1: Reduce noise
img_smooth = img.smooth(sigma=0.2)
img_np = img_smooth.A
```

*Figure 2: Code to minimize noise using function smooth().*

## Conversion to HSV Color Space

Since the RGB space does not separate color information well from illumination, the image is converted to HSV space (see Figure 3). This allows working specifically on the hue channel, facilitating detection of specific colors regardless of light intensity.

```
# Step 2: Convert to HSV color space
hsv_img = cv.cvtColor(img_np, cv.COLOR_RGB2HSV)
```

*Figure 3: Code to convert the image from RGB to HSV.*

## Color Segmentation and Morphological Processing

A range of hue, saturation, and value (HSV) values corresponding to the characteristic brown stem color is defined. Using the `cv2.inRange()` function, a **binary mask** is generated where pixels within that range are white, and the rest are black.

To eliminate noise and connect fragmented regions, **morphological erosion and dilation** operations are applied using a structuring kernel (see Figure 4). These operations improve the shape and continuity of detected regions.

After these steps, a processed binary mask is obtained (Figure 5), where white areas represent pixels identified as brown and black areas represent the background or irrelevant parts of the scene.

```
# Step 3: Define the color range for brown detection
lower_brown = np.array([5, 50, 20])
upper_brown = np.array([30, 255, 200])

# Step 4: Create a binary mask for the brown color
mask_brown = cv.inRange(hsv_img, lower_brown, upper_brown)

# Step 5: Reduce noise with morphological operations
kernel = np.ones((5, 5), np.uint8)
mask_brown = cv.erode(mask_brown, kernel, iterations=1)
mask_brown = cv.dilate(mask_brown, kernel, iterations=2)
```

*Figure 4: Definition of value range for brown color, application of binary mask, and noise reduction.*

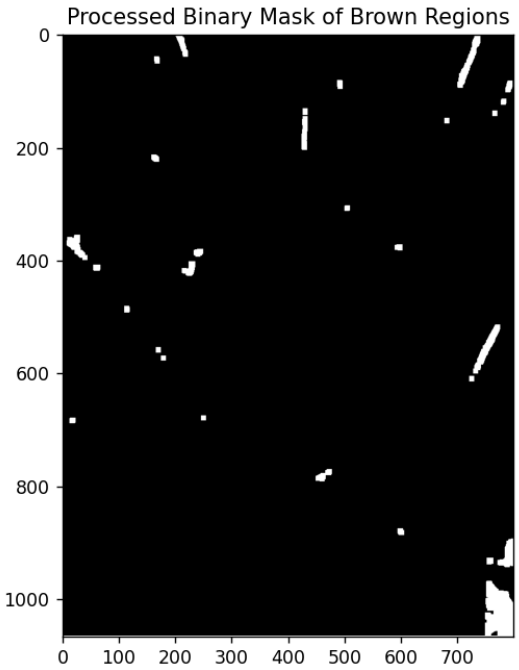


*Figure 5: Binary mask of brown regions.*

## Blob Detection, Filtering, and Centroid Extraction

The binary mask is converted into an Image object from the machinevisiontoolbox library, and blob detection algorithms are applied to find connected regions. Each blob represents a possible stem location. To avoid errors, blobs are filtered by minimum and maximum area, and those with invalid moments are discarded. From valid blobs, **centroids** are extracted, representing the estimated stem position (see Figure 6).

```
# Step 6: Convert to an mvt.Image object for blob detection
masked_image_obj = mvt.Image(mask_brown)

try:
    # Step 7: Detect blobs without setting a min/max area (filter manually)
    blobs = masked_image_obj.blobs()

    # Step 8: Manually filter blobs by area (ignore very small or very large blobs)
    valid_blobs = [blob for blob in blobs if 50 < blob.area < 50000 and blob.moments[0] > 0]

    # Debug: Print the number of valid blobs detected
    print(f"Number of valid blobs detected: {len(valid_blobs)}")

    # Step 9: Check if any valid blobs were found
    if len(valid_blobs) == 0:
        print(" No valid blobs detected. Check the binary mask segmentation.")
        return []

    # Step 10: Collect the centroids of the detected blobs
    results = [blob.centroid for blob in valid_blobs]
    return results
```

*Figure 6: Blob detection, area filtering and extraction of valid centroids.*

## Result Visualization

Finally, the detected centroids are marked on the original image using colored circles, as shown in Figure 7. This allows visual validation of the algorithm's accuracy.

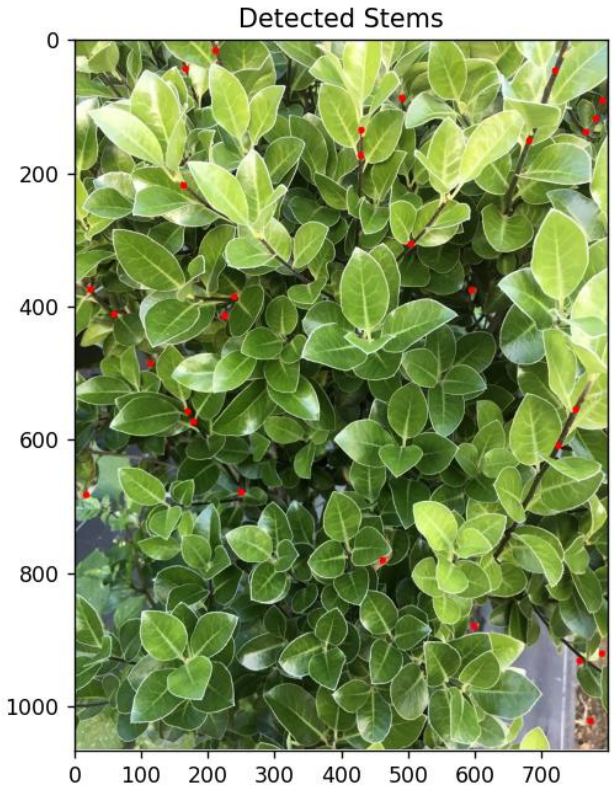


*Figure 7: Final solution showing the found stems with a red circle.*

# Future Development Possibilities

The developed algorithm represents an important first step toward creating **autonomous robotic systems** capable of interacting with natural environments. These types of solutions can evolve to be integrated into robots that not only detect stems but also perform tasks such as automated branch cutting or crop maintenance, which would have valuable applications in precision agriculture and forestry work.

Furthermore, the system can be expanded to work with **video sequences** instead of static images, enabling real-time use and improving its applicability in mobile robots. Continuous image processing would allow the robot to make dynamic decisions based on its perceived environment. Overall, this type of artificial vision algorithm forms the basis for more complex developments and represents a fundamental component in intelligent systems in the field of robotics.

# Conclusion

This project successfully implemented a functional computer vision system capable of detecting stems in images using color segmentation and blob analysis. By leveraging tools available in Python and solid foundations of computer vision, an efficient and modular algorithm was developed that accurately identifies regions of interest in controlled environments. The approach based on the HSV color space, along with morphological preprocessing and connected region filtering, yielded reliable visual results aligned with the project's objective.

Beyond its current functionality, this work represents a solid foundation for more advanced developments in autonomous robotics. The concepts and techniques implemented can be scaled up to systems that process real-time video, operate under varying environmental conditions, or even interact physically with their surroundings. Altogether, this project demonstrates how a technically well-grounded solution can significantly contribute to the advancement of intelligent robotic systems.